1. **Scope của Variable**

Là phạm vi mà ta có thể sử dụng 1 variable

1. **Variable**

Có 2 loại là:

* **User-defined variable**

User-defined variable chỉ có thể là local hoặc session

* + **Local**

Là biến có scope chỉ sử dụng được ở trong function, store procedure, trigger.

Cú pháp: DECLARE variable\_name = value;

* + **Global**

Là biến có scope chỉ sử dụng được ở trong toàn bộ SQL

Cú pháp:

Cách 1: SET @@global.variable\_name = value;

Cách 2: SET GLOBAL variable\_name = value;

* + **Session**

Là biến có scope chỉ sử dụng t

Cú pháp:

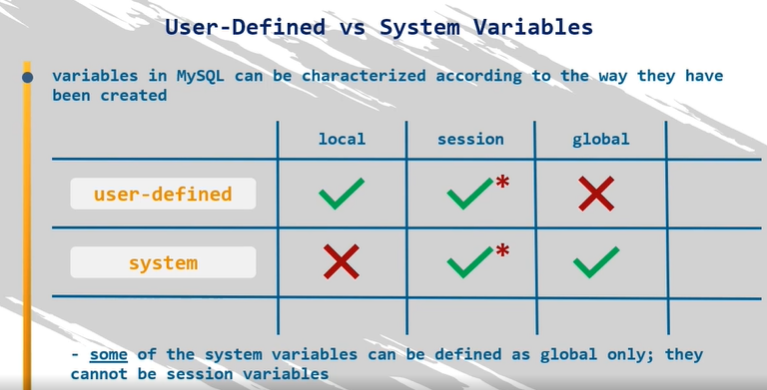
Cách 1: SET @variable\_name = value;

Cách 2: SET SESSION variable\_name = value;

* **System variable**

Là biến có scope chỉ sử dụng được ở trong toàn bộ SQL (do SQL tạo ra)

System variable chỉ có thể là session hoặc global

****

* Phân biệt session và connection

1. **Trigger**

* Tự động thực thi các lệnh SQL hoặc Store Procedure BEFORE | AFTER các lệnh INSERT, UPDATE, hoặc DELETE.
* Là 1 dạng đặc biệt của Store Procedure
* Sử dụng để
  + Kiểm tra tính toàn vẹn, bắt lỗi logic của cơ sở dữ liệu
  + Các ràng buộc Contraint, giá trị default phức tạp
  + Kiểm soát những thay đổi của dữ liệu trong table
* Cách hoạt động
  + **MySQL**
    - **Before**
      * Trigger được gọi **trước câu** khi câu insert xảy ra
      * Có thể truy cập và thay đổi được value ở trong NEW table

<https://www.mysqltutorial.org/mysql-triggers/mysql-before-insert-trigger/>

* + - **After**
      * Trigger được gọi **sau câu** khi câu insert xảy ra
      * Có thể truy cập nhưng không thể thay đổi được value ở trong NEW table

<https://www.mysqltutorial.org/mysql-triggers/mysql-after-insert-trigger/>

* + - **New table**

Trở tới data **sau khi** thay đổi (dữ liệu trong câu insert)

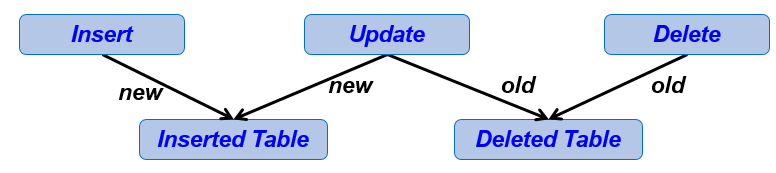
* + - **Old table**

Trở tới data **trước khi** thay đổi (dữ liệu trong real table)

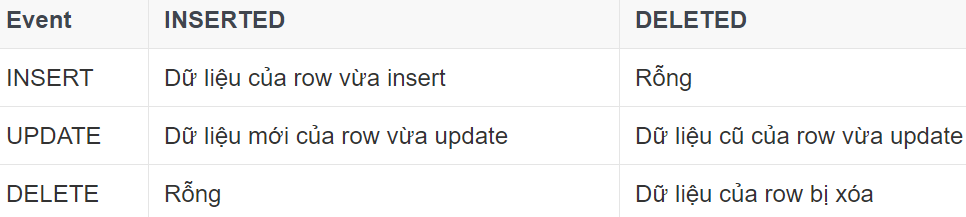
VD:

https://viblo.asia/p/trigger-in-mysql-l0rvmm5AvyqA

* + **SQL Server**



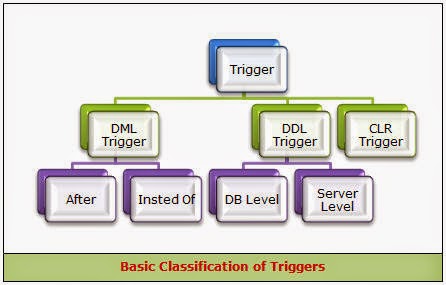
* + - **FOR | AFTER** (Default Trigger)
      * Thực hiện Trigger sau khi đã thay đổi data thành công ở Table Inserted và Deleted
      * Insert 🡪 Table Inserted, Deleted 🡪 Check Statement Trigger 🡪 Khi Statement thành công thì sẽ insert vào table thực
      * Không sử dụng được cho View
    - **INSTEAD OF** 
      * Thực hiện Trigger, không thực hiện thay đổi data (thay thế lệnh thay đổi của sql bằng trigger)
      * Khi thực hiện thì dữ liệu sẽ tác động vào Table Inserted hoặc Deleted và không tác động vào database thực (chỉ có trigger tác động được vào database thực)
      * Sử dụng được cho View



* + - Khi Created Trigger thì SQL sẽ cấp quyền cho mình truy cập tới Temporary tables Inserted và Deleted
      * Tables Inserted và Deleted có cấu trúc giống với table đang thực thi và được lưu ở Memory (table thực được lưu ở disk)
      * 2 Tables chỉ được cấp quyền truy cập khi Trigger chạy

🡺 Không truy cập được Inserted và Deleted ở ngoài Trigger

* + - Khi thực hiện INSERT | UPDATE | DELETE thì data sẽ được thêm vào 2 tables Inserted và Deleted
* Có 2 loại



* + **DML TRIGGER** (Standard triggers) (*INSERT, UPDATE, DELETE*)
  + **DDL TRIGGER** (CREATE, ALTER, DROP…)
* Có thể disable hoặc enable trigger

Default khi create là enable

* Cú pháp
  + Create Trigger
    - DML

DROP TRIGGER IF EXISTS *trigger\_name();*

DELIMITER $$

CREATE TRIGGER *trigger\_name()*

BEFORE/AFTER/INSTEAD OF INSERT/UPDATE/DELETE ON *name\_table*

FOR EACH ROW

BEGIN

-- logic

SELECT ...;

END $$

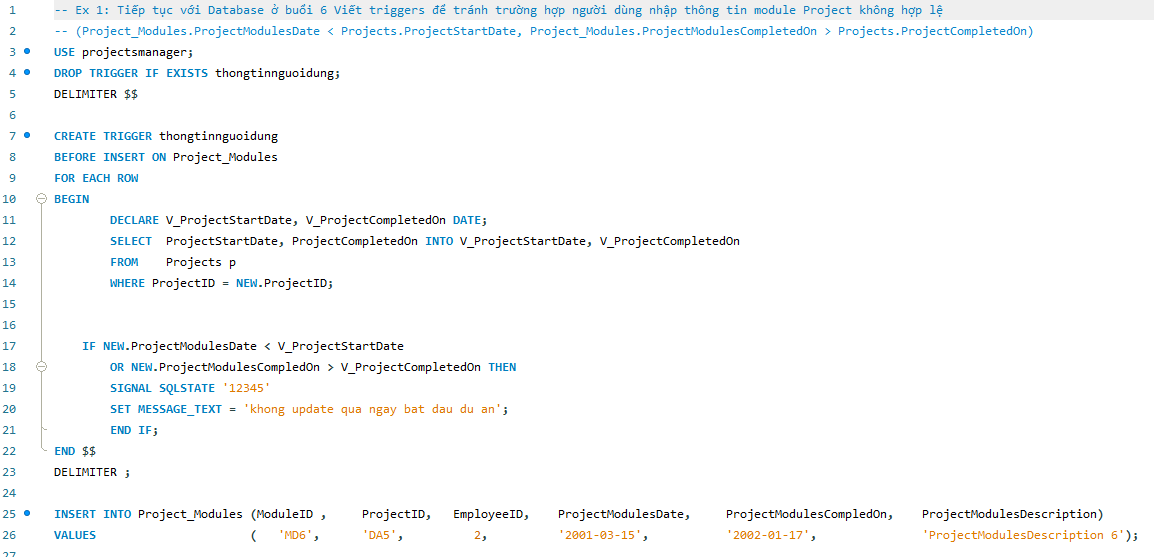
DELIMITER ;

* + - DDL
  + Alter Trigger
  + DISABLE/ENABLE Trigger
  + Drop trigger

DROP TRIGGER IF EXISTS *trigger\_name();*

* + Call
    - Auto call.

VD:



1. **Index**

Giống mục lục của quyển sách.

Cú pháp:

CREATE INDEX *index\_name*

ON *table\_name* (*column\_name\_1*, *column\_name\_2*);

1. **CASE WHEN**

Giống switch case ở Java, C, C++

Dùng để lọc các điều kiện

Có thể dùng ở trong SELECT

VD:

